A

Patent Application
Docket No. 33361-00028USPT
STA.WSL.025

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

STEVEN EUGENE LOVETTE

For: TECHNIQUE FOR DETECTING CORRUPTION ASSOCIATED WITH A STACK IN A STORAGE DEVICE

BOX APPLICATION
Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

CERTIFICATE OF MAILING BY EXPRESS MAIL

"EXPRESS MAIL" Mailing Label No. EL123353715US
Date of Deposit November 23, 1998
I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

Type or Print Name CAROL MARSTALLER

Signature

Dear Sir:

REQUEST FOR FILING A NATIONAL PATENT APPLICATION

Transmitted herewith for filing, please find the following:

_X_ 1. Specification, claims and abstract of the above-referenced patent application having 24 pages.

_X_ 2. _2_ pages of drawing(s) (Figs. 1-4) (___ formal / _X_ informal).

_X_ 3. Declaration and Power of Attorney (signed)

____ 3A. No filing fee, Oath, or Declaration is enclosed pursuant to 35 U.S.C. 53(d).

___ 4. Power of Attorney By Assignee (___ signed ___ unsigned).

____ 5. Information Disclosure Statement along with Form PTO-1449 and references.

____ 6. This is a: ____ CIP, ____ DIV, ____ CONT, or ____ substitute Application (MPEP 201.09) of Application Serial No. ____ filed ____; or, is a ____ reissue of U.S. Patent No. ____ filed on ____.

An extension to extend the life of the above prior Application to at least the date of filing hereof
(One box must be marked)
(a)____ is concurrently being filed in that prior Application,
(b)____ was previously filed in that prior Application (check length of prior extension),
(c)____ is not necessary for copendency (double check before X'ing this).

_x_ 7. Attached is an assignment to SAMSUNG ELECTRONICS CO., LTD. Please return the recorded assignment to the undersigned. (NOTE: add recordal fee below).

____ 8. Priority is claimed under 35 U.S.C. § 119 based on filing in _____.

	Provisional Application No.	Filing Date
(1)		

____ (No.) Certified copy (copies) ____ are attached; or ____ were previously filed on ____.

____ 9. Attached: ____ (No.) verified statement(s) establishing "small entity" status under 37 CFR § 1.9 and 1.27.

_X_ 10. Attached:

_X_ Return Postcard

____ 11. Preliminary Amendment:

Prior to a first Office Action, kindly amend the Application as follows:

12. The following Filing Fee calculation is based on the claims filed less any claims canceled by the Preliminary Amendment of Item 11.

					SMALL ENTITY RATE		LARGE ENTITY RATE		
BASIC FEE					$395	OR	$790	=	$790.00
	NUMBER FILED			NUMBER EXTRA					
TOTAL CLAIMS	25	-20	=	5 (at least 0)	x 11	OR	x 22	=	+$ 110.00
INDEP. CLAIMS	3	- 3	=	___ (at least 0)	x 41	OR	x 82	=	+$ 0
If any proper multiple dependent claim (ignore improper) is present (Enter $0.00 if this is a reissue application.)					+$130	OR	+$260	=	+$ 0
If assignment is x'd (line 7), add recording fee $40.00									+$ 40.00
Attached is a Rule 47 Petition (inventor refuses to sign or cannot be reached) $130									+$___
TOTAL FILING FEE									**=$ 940.00**

X 13. A check in the amount of $940.00 for the filing fees for the application and a check for $40.00 for the Assignment calculated in Item 12 is attached. Please charge any deficiency or credit any overpayment to Jenkens & Gilchrist's Deposit Account No. 10-0447.

____ 14. Please charge Deposit Account No. 10-0447 in the amount of $_____ to cover the Filing Fee calculated in Item 12. This sheet is attached in duplicate.

X 15. The Commissioner is hereby authorized to charge any fee specifically authorized hereafter, or any missing or insufficient fee(s) filed, or asserted to be filed, or which should have been filed herewith or concerning any paper filed hereafter, and may be required under 37 CFR 1.16-1.18 (missing or insufficiencies only) now or hereafter relative to this application and for the resulting Official Document under 37 CFR 1.20, OR credit any overpayment to JENKENS & GILCHRIST'S Deposit Account No. 10-0447, **for which purpose a duplicate copy of this sheet is attached.**

Patent Application
Docket No. 33361-00028USPT
STA.WSL.025

The Commissioner is not authorized to charge the issue fee until/unless an issue fee transmittal form is filed.

Respectfully submitted,

JENKENS & GILCHRIST, P.C.

By: ______________________
Name: Steven R. Greenfield
Reg. No. 38,166

Date: November 23, 1998

Jenkens & Gilchrist, P.C.
1445 Ross Avenue, Suite 3200
Dallas, Texas 75202
(214) 855-4789
(214) 855-4300 (fax)

Patent Application
Docket No. 33361-00028USPT

CERTIFICATE OF MAILING BY EXPRESS MAIL

"EXPRESS MAIL" Mailing Label No. EL133553[illegible]US
Date of Deposit: 11/23/98
I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Box Patent Application, Washington, D.C. 20231

Type or Print Name: CAROL MARSTALLER

Carol S. Marstaller
Signature

TECHNIQUE FOR DETECTING CORRUPTION ASSOCIATED WITH A STACK IN A STORAGE DEVICE

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to multi-tasking systems and, more particularly, to a technique for detecting corruption associated with a stack in a storage device.

Background of the Present Invention

A real time embedded system often provides a multi-tasking environment in order to meet diverse application requirements. In a multi-tasking environment, an individual stack (i.e., an individual work space) is typically required for each task. Each stack provides a location, or a group

of locations, where dynamic function variables may be stored as needed for a corresponding task. Typically, each stack is allocated a dedicated range of sequential memory, although a stack can also be allocated a dedicated space of non-sequential memory. Regardless, stacks have always had problems in the areas of overflow and underflow.

Stack overflow occurs when the stack memory is insufficient to meet workload demand. Stack underflow occurs when a task attempts to pop too many function variables off of the stack. The net result of either operation is the corruption of adjacent memory. Corruption of adjacent memory results in non-predictable behavior and difficult to impossible casual analysis of the problem (i.e., finding the source of the problem).

Traditional solutions to stack overflow and underflow problems are to either significantly over-allocate stack memory or use hardware write protect schemes. The first solution basically ignores the problem, hoping but not ensuring that it never happens. Since stack corruption is a dynamic condition, this solution is fraught with shortcomings. The second solution solves the detection

problem, but at the cost of additional hardware complexity and corresponding hardware failure rates.

A software solution to the above-described stack overflow and underflow problems is to keep track of the number of function variables that are on the stack by adding one (i.e., +1) to a counter whenever a push operation occurs, and by subtracting one (i.e., -1) from the counter whenever a pop operation occurs. However, whenever a complex push/pop operation is performed (i.e., when push/pop operations are performed in a number of different places), it is difficult to ensure that all such places perform the requisite addition/subtraction operation. As an alternative, a centralized procedure and/or function for performing the push and pop operations can be created, wherein the appropriate addition and subtraction operations are performed therein. However, this requires calling such procedures and/or functions, which requires additional memory space and processor time.

In view of the foregoing, it would be desirable to provide a technique for overcoming the above-described stack overflow and underflow problems, while also overcoming the shortcomings of the above-described prior art solutions.

More particularly, it would be desirable to provide a technique for detecting stack corruption in a multi-tasking environment.

Objects of the Present Invention

The primary object of the present invention is to provide a technique for detecting corruption associated with a stack in a storage device.

The above-stated primary object, as well as other objects, features, and advantages, of the present invention will become readily apparent from the following detailed description which is to be read in conjunction with the appended drawings.

SUMMARY OF THE INVENTION

According to the present invention, a technique for detecting corruption associated with a stack in a storage device is provided. The technique is beneficially utilized in a multi-tasking environment wherein a processor typically performs a task by retrieving a message from a message queue and processing the message by calling, or invoking, one or more specific functions which are required to perform the

task based upon information contained in the message. For each task there is a stack formed in a storage device, for example a memory device, for storing function variables as needed for performing the corresponding task. The corruption associated with the stack can arise as a result of a faulty task, function, or the stack itself.

The technique can be realized by having a processing device such as, for example, a digital microprocessor, insert a quantity of information adjacent to the stack in the storage device. The quantity of information has an initial state which can represent a variety of predetermined configurations such as, for example, a bit pattern, a processor readable address, or a processor readable instruction. After the quantity of information has been inserted adjacent to the stack in the storage device, the processing device inspects the quantity of information after certain operations occur so as to identify any deviation from the initial state and thereby detect corruption associated with the stack in the storage device.

In a first operation, data is added to the stack after the quantity of information has been inserted adjacent to the stack in the storage device. This first operation, which is

typically referred to as a push operation, can cause the quantity of information to deviate from the initial state. If upon inspection such a deviation is identified, then the push operation is recorded and remedial measures can be taken.

In a second operation, data is removed from the stack after the quantity of information has been inserted adjacent to the stack in the storage device. This second operation, which is typically referred to as a pop operation, can also cause the quantity of information to deviate from the initial state. If upon inspection such a deviation is identified, then the pop operation is recorded and remedial measures can be taken.

After a deviation from the initial state has been identified, the processing device typically restores the quantity of information to the initial state, thereby allowing the detection of any subsequent deviations from the initial state.

In one aspect of the present invention, the processing device inserts a first quantity of information adjacent to a top of the stack in the storage device, and inserts a second quantity of information adjacent to the bottom of the

stack in the storage device. This aspect of the present invention allows both stack overflow and stack underflow conditions to be detected.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to facilitate a fuller understanding of the present invention, reference is now made to the appended drawings. These drawings should not be construed as limiting the present invention, but are intended to be exemplary only.

Figure 1 is a schematic diagram of a processing system for facilitating the implementation of a multi-tasking environment in accordance with the present invention.

Figure 2 is an illustration of an exemplary message queue for storing a plurality of messages in the order they were received and/or in the order of occurrence of their corresponding events in accordance with the present invention.

Figure 3 is an illustration of an exemplary stack structure wherein a task is allocated a stack space in memory for storing variables associated with the different functions that are invoked by the task in accordance with the present invention.

Figure 4 is an illustration of an exemplary stack structure having a plurality of stacks and a corresponding plurality of guard frames in accordance with the present invention.

DETAILED DESCRIPTION OF A PREFERRED EXEMPLARY EMBODIMENT

Referring to Figure 1, there is shown a processing system 10 including at least one processor (P) 12, memory (M) 14, and input/output (I/O) interface 16, connected to each other by a bus 18, for facilitating the implementation of a multi-tasking environment in accordance with the present invention. In such a multi-tasking environment, the processor 12 typically performs a task by retrieving a message from a message queue and processing the message by calling, or invoking, one or more specific functions which are required to perform the task based upon information contained in a message. A message typically identifies an event which is received by the processing system 10 through the input/output interface 16. Alternatively, a message can identify an event which occurs within the processing system 10. In any case, each message is typically stored in the message queue in the order it is received or in the order of

occurrence of the corresponding event, which are often one in the same. For example, referring to Figure 2, there is shown a message queue 20 containing a plurality of messages 22 which are stored in the order they were received and/or in the order of occurrence of their corresponding events. The message queue is typically located in the memory 14.

Referring again to Figure 1, the memory 14 also typically contains operating system software which, when the processing system 10 is initialized, causes the processor 12 to establish a plurality of tasks and allocate a corresponding plurality of dedicated work spaces, or stacks, in the memory 14. Each stack provides a location, or a group of locations, in the memory 14 where function variables may be stored as needed for a corresponding task. For example, referring to Figure 3, there is shown a stack structure 30 wherein each task is allocated a stack space in the memory 14 for storing variables associated with the different functions that are invoked for each task. It should be noted that, although Figure 3 shows each stack as encompassing a dedicated range of sequential memory, the present invention allows a stack to encompass a dedicated space of non-sequential memory, as described in detail below.

Referring again to Figure 1, each task is essentially a collection of functions which are typically established along with their corresponding tasks when the processing system 10 is initialized. Each function typically has a predefined template that includes calling the function, storing function parameters, and local variables. The number of times each function may be called is unknown. The number of functions that may be called is also unknown. The combination of these two factors make the stack size unknown. This is explained on the next page. These memory requirements are typically not fully known at compile time. Thus, due to this lack of knowledge, the amount of stack space that is allocated for each task when the processing system 10 is initialized is often inadequate, and stack overflow and underflow can easily occur without safeguards.

In operation, the operating system software causes the processor 12 to activate a specific task based upon workload demands and priorities. The task begins when the processor 12 retrieves the first message from the message queue and begins processing the message. The processor 12 processes the message by invoking one or more specific functions which are required to perform the task based upon information

contained in the message. As each function is invoked, additional stack space is used.

The dynamic aspect of a stack is its depth. That is, the required depth of a stack depends upon how many function variables will be placed on the stack for any single execution thread as function variables are pushed on and popped off the stack in order to process a message. This is non-deterministic in nature due to implementation techniques such as recursion. Thus, due to the non-deterministic nature of the required depth of a stack, the amount of stack space that is allocated for each task when the processing system 10 is initialized is often inadequate, and stack overflow and underflow can easily occur without safeguards.

In accordance with the exemplary embodiment of the present invention, safeguards are employed so as to avoid stack overflow and underflow. That is, a guard frame is inserted at the top and bottom of each stack so as to allow stack corruption caused by stack overflow and underflow to be detected by a guard function. Once the stack corruption is detected, the function which caused the stack corruption to occur can be determined and remedial measures can be taken.

Referring to Figure 4, there is shown a stack structure 40 having a plurality of stacks 42 and a corresponding plurality of guard frames 44 in accordance with the present invention. Each guard frame 44 borders a corresponding stack 42 and preferably comprises a fixed bit pattern that is stored as one or more bytes in the memory 14. For example, a guard frame 44 can be formed of a sequence of bytes containing a bit pattern of alternating ones (i.e., logic level 1) and zeros (i.e., logic level 0). Alternatively, a guard frame 44 can be formed of a sequence of bytes containing an address or instruction which causes the processor 12 to reach some predetermined location or state.

The size of each guard frame 44 is typically implementation dependent, but is always relatively small in comparison to a corresponding stack 42. The key issues are that each guard frame 44 should be large enough to be unique from random data patterns, large enough to provide corruption protection to an adjacent stack, and small enough to not be a performance burden. At this point it should be noted that two physically adjacent stacks (e.g., stack 1 and stack 2 in Figure 4) may share the same guard frame (e.g., the guard frame 44 located between stack 1 and stack 2 in Figure 4). However, if two stacks are not physically adjacent (i.e., the

two stacks are separated by memory space that is dedicated to another purpose), then a separate guard frame will typically be required for each stack (i.e., the two stacks cannot share a guard frame). For example, if the memory space 46 between stack 2 and stack N was dedicated to another purpose (i.e., memory space 46 was not dedicated to one or more stacks), then there would be no stacks adjacent to either stack 2 or stack N in memory space 46 and neither stack 2 nor stack N could share a guard frame with another stack in memory space 46.

Each time a function variable is pushed on or popped off a stack 42, the appropriate guard frame 44 is checked by a guard function. For example, when a function variable is pushed onto a stack 42, the guard function checks the guard frame 44 at the bottom of the stack 42 to see if an overflow has occurred or might occur based on the minimum space requirements of the function. An overflow occurs when the guard frame 44 has been corrupted (e.g., overwritten) or insufficient memory exists to satisfy the minimum space requirements of the function. If the guard function determines that an overflow has occurred, or might occur, then the guard function records the offending function and task and begins task cleanup and recreation. As part of task

cleanup, the guard frame 44 is restored to its pre-corrupted state.

In the opposite case, when a function variable is popped off of a stack 42, the guard function checks the guard frame 44 at the top of the stack 42 to see if an underflow has occurred. An underflow occurs when there are no function variables on the stack 42 and a pop operation is performed In such a case the guard frame 44 would be corrupted. This typically occurs when there is some sort of corruption of the task, the function, or even the stack 42 which causes stack depth confusion. If the guard function determines that an underflow has occurred, then the guard function records the offending function and task and begins task cleanup and recreation. Again, as part of task cleanup, the guard frame 44 is restored to its pre-corrupted state.

At this point it should be noted that the guard function is typically a separate function which is invoked whenever a push or pop operation occurs. That is, the guard function, in conjunction with the guard frame, is a software-based solution to stack overflow and underflow problems which overcomes the shortcomings of traditional solutions such as excess memory allocation and hardware write protect schemes. Furthermore, unlike a prior art software solution, the

present invention guard function and guard frame solution does not require a counter to be maintained or a computations to be performed. That is, the present invention guard function and guard frame solution requires minimal overhead.

The present invention guard function and guard frame solution to stack overflow and underflow problems provides a mechanism for detecting stack corruption, preventing corruption of adjacent stacks, and isolating offending functions, tasks, and software. By implementing this mechanism, the reliability of real time embedded controllers is improved. An additional advantage of this solution is that it can be applied to existing systems that lack hardware write protect schemes and to systems that require memory mining.

The present invention is not to be limited in scope by the specific embodiments described herein. Indeed, various modifications of the present invention, in addition to those described herein, will be apparent to those of skill in the art from the foregoing description and accompanying drawings. Thus, such modifications are intended to fall within the scope of the appended claims.

WHAT IS CLAIMED IS:

1. A method for detecting corruption associated with a stack in a storage device, the method comprising the steps of:

inserting a quantity of information adjacent to the stack in the storage device, the quantity of information having an initial state; and

inspecting the quantity of information so as to identify any deviation from the initial state and thereby detect corruption associated with the stack in the storage device.

2. The method as defined in claim 1, wherein the initial state of the quantity of information represents a bit pattern.

3. The method as defined in claim 1, wherein the initial state of the quantity of information represents a processor readable address.

4. The method as defined in claim 1, wherein the initial state of the quantity of information represents a processor readable instruction.

5. The method as defined in claim 1, further comprising the step of:

adding data to the stack after inserting the quantity of information adjacent to the stack in the storage device.

6. The method as defined in claim 5, wherein the data is added during a push operation, further comprising the step of:

recording the push operation after identifying any deviation from the initial state.

7. The method as defined in claim 1, further comprising the step of:

removing data from the stack after inserting the quantity of information adjacent to the stack in the storage device.

8. The method as defined in claim 7, wherein the data is removed during a pop operation, further comprising the step of:

recording the pop operation after identifying any deviation from the initial state.

9. The method as defined in claim 1, further comprising the step of:

restoring the quantity of information to the initial state after identifying any deviation from the initial state.

10. The method as defined in claim 1, wherein the step of inserting the quantity of information adjacent to the stack in the storage device includes:

inserting a first quantity of information adjacent to a top of the stack in the storage device; and

inserting a second quantity of information adjacent to a bottom of the stack in the storage device.

11. A system for detecting corruption associated with a stack in a storage device, the system comprising:

a processor; and

a storage medium for storing instructions that are readable by the processor and thereby cause the processor to operate so as to:

insert a quantity of information adjacent to the stack in the storage device, the quantity of information having an initial state; and

inspect the quantity of information so as to identify any deviation from the initial state and thereby detect corruption associated with the stack in the storage device.

12. The system as defined in claim 11, wherein the initial state of the quantity of information represents a bit pattern.

13. The system as defined in claim 11, wherein the initial state of the quantity of information represents a processor readable address.

14. The system as defined in claim 11, wherein the initial state of the quantity of information represents a processor readable instruction.

15. The system as defined in claim 11, further causing the processor to operate so as to:

add data to the stack after inserting the quantity of information adjacent to the stack in the storage device.

16. The system as defined in claim 15, wherein the data is added during a push operation, further causing the processor to operate so as to:

record the push operation after identifying any deviation from the initial state.

17. The system as defined in claim 11, further causing the processor to operate so as to:

remove data from the stack after inserting the quantity of information adjacent to the stack in the storage device.

18. The system as defined in claim 17, wherein the data is removed during a pop operation, further causing the processor to operate so as to:

record the pop operation after identifying any deviation from the initial state.

19. The system as defined in claim 11, further causing the processor to operate so as to:

restore the quantity of information to the initial state after identifying any deviation from the initial state.

20. The system as defined in claim 11, further causing the processor to operate so as to:

insert a first quantity of information adjacent to a top of the stack in the storage device; and

insert a second quantity of information adjacent to a bottom of the stack in the storage device.

21. A computer system including a mechanism for detecting corruption associated with a stack in a storage device, the computer system comprising:

a computer readable storage medium; and

computer programming stored on the storage medium; wherein the stored computer programming is configured to be readable from the computer readable storage medium by one or more computers and thereby cause the one or more computers to operate so as to:

insert a quantity of information adjacent to the stack in the storage device, the quantity of information having an initial state; and

inspect the quantity of information so as to identify any deviation from the initial state and thereby detect corruption associated with the stack in the storage device.

22. The computer system as defined in claim 21, wherein the initial state of the quantity of information represents a bit pattern.

23. The computer system as defined in claim 21, wherein the initial state of the quantity of information represents a processor readable address.

24. The computer system as defined in claim 21, wherein the initial state of the quantity of information represents a processor readable instruction.

25. The computer system as defined in claim 21, further causing the one or more computers to operate so as to:

add data to the stack after inserting the quantity of information adjacent to the stack in the storage device.

ABSTRACT OF THE DISCLOSURE

A technique for detecting corruption associated with a stack in a storage device is disclosed. In one embodiment, the technique is realized by having a processing device insert a quantity of information adjacent to the stack in the storage device, wherein the quantity of information has an initial state. The processing device then inspects the quantity of information so as to identify any deviation from the initial state and thereby detect corruption associated with the stack in the storage device.

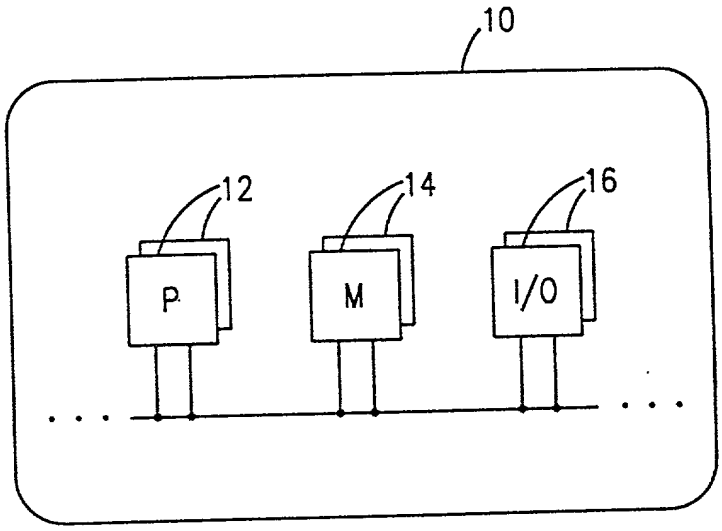
10
12
14
16
P
M
I/O

FIG. 1

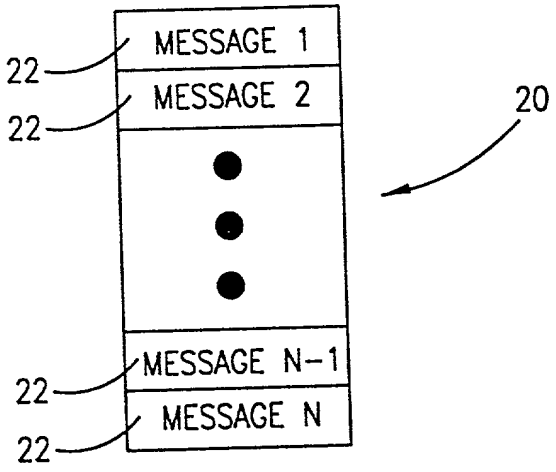
MESSAGE 1
MESSAGE 2
MESSAGE N-1
MESSAGE N
22
22
22
22
20

FIG. 2

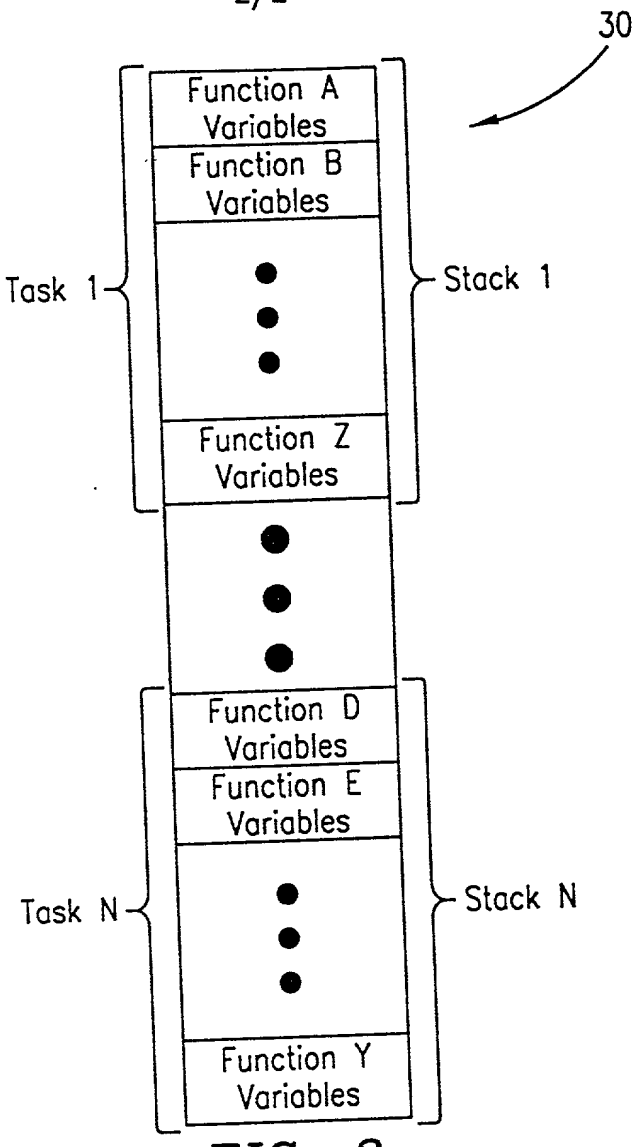
30
Function A Variables
Function B Variables
Task 1
Stack 1
Function Z Variables
Function D Variables
Function E Variables
Task N
Stack N
Function Y Variables

FIG. 3

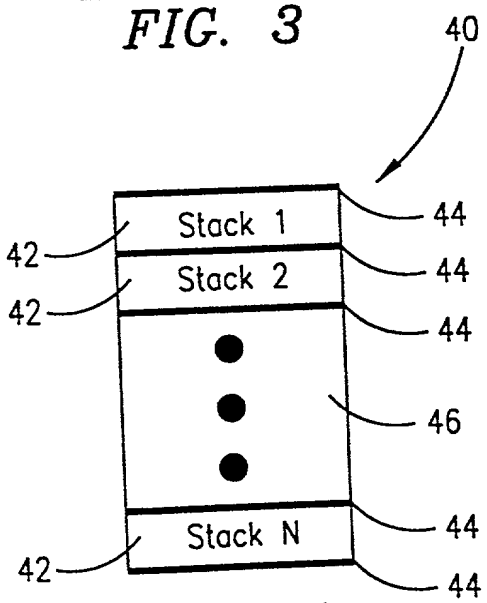
40
Stack 1
Stack 2
Stack N
42
44
46

FIG. 4

STA.WSL.025 PATENT APPLICATION
33361-00028USPT

RULES 63 AND 67 (37 C.F.R. 1.63 and 1.67)
DECLARATION AND POWER OF ATTORNEY

FOR UTILITY/DESIGN/CIP/PCT NATIONAL APPLICATIONS

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name; and

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **TECHNIQUE FOR DETECTING CORRUPTION ASSOCIATED WITH A STACK IN A STORAGE DEVICE,** the specification of which: (mark only one)

__X__ (a) is attached hereto.
_____ (b) was filed on _____ as Application Serial No. _____ and was amended on _____ (if applicable)
_____ (c) was filed as PCT International Application No. PCT/_____ on ____ and was amended on _____ (if applicable).
_____ (d) was filed on _____ as Application Serial No. ____ and was issued a Notice of Allowance on _____.
___ (e) was filed on _____and bearing attorney docket number _____

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as amended by any amendment referred to above or as allowed as indicated above.

I acknowledge the duty to disclose all information known to me to be material to the patentability of this application as defined in 37 CFR § 1.56. If this is a continuation-in-part (CIP) application, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose to the Office all information known to me to be material to patentability of the application as defined in 37 CFR § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this CIP application.

I hereby claim foreign priority benefits under 35 U.S.C. § 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate filed by me or my assignee disclosing the subject matter claimed in this application and having a filing date (1) before that of the application

on which my priority is claimed or, (2) if no priority is claimed, before the filing date of this application:

PRIOR FOREIGN PATENTS

Number	Country	Month/Day/Year Filed	Date first laid-open or Published	Date patented or Granted	Priority Claimed Yes	No

I hereby claim the benefit under 35 U.S.C. § 120/365 of any United States application(s) listed below and PCT international applications listed above or below:

PRIOR U.S. OR PCT APPLICATIONS

Application No. (series code/serial no.)	Month/Day/Year Filed	Status(pending, abandoned, patented)

I hereby appoint JOHN HAN, Reg. No. 41,403; THOMAS L. CRISMAN, Reg. No. 24,846; THOMAS L. CANTRELL, Reg. No. 20,849; STANLEY R. MOORE, Reg. No. 26,958, H. MATHEWS GARLAND, Reg. No. 19,129; GERALD T. WELCH, Reg. No. 30,332; ROGER L. MAXWELL, Reg. No. 31,855; P. WESTON MUSSELMAN, JR., Reg. No. 31,644; JERRY R. SELINGER, Reg. No. 26,582; J. KEVIN GRAY, Reg. No. 37,141; STEVEN R. GREENFIELD, Reg. No. 38,166; ANDRE M. SZUWALSKI, Reg. No. 35,701; STUART D. DWORK, Reg. No. 31,103; CRAIG A. HOERSTEN, Reg. No. 38.917; RICHARD J. MOURA, Reg. No. 34,883; RICHARD A. MYSLIWIEC, Reg. No. 40,098; RAYMOND VAN DYKE, Reg. No. 34,746; BRIAN D. WALKER, Reg. No. 37.751; and J. PAT HEPTIG, Reg. No. 40,643, all of the firm of **JENKENS & GILCHRIST, P.C.**, 3200 Fountain Place, 1445 Ross Avenue, Dallas, Texas 75202-2799, as my attorneys and/or agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith, and to file and prosecute any international patent application filed thereon before any international authorities under the Patent Cooperation Treaty, and I hereby authorize them to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/organization who/which first sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct them in writing to the contrary.

Please address all correspondence and direct all telephone calls to:

Steven R. Greenfield
Jenkens & Gilchrist, A Professional Corporation
3200 Fountain Place
1445 Ross Avenue
Dallas, Texas 75202-2799
214/855-4789 Fax: 214/855-4300

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

NAMED INVENTOR(S)

1	Steven Eugene Lovette **Full Name**	Steven Eugene Lovette **Inventor's Signature**	11/23/98 **Date**
	1068 Sunset Court Keller, Texas 76248 **Residence** (city, state, country)		USA **Citizenship**
	1068 Sunset Court Keller, Texas 76248 **Post Office Address** (include zip code)		

(FOR ADDITIONAL INVENTORS, check here ____ and add additional sheet for inventor information regarding signature, name, date, citizenship, residence and address)